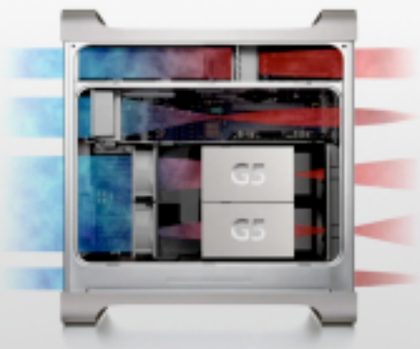
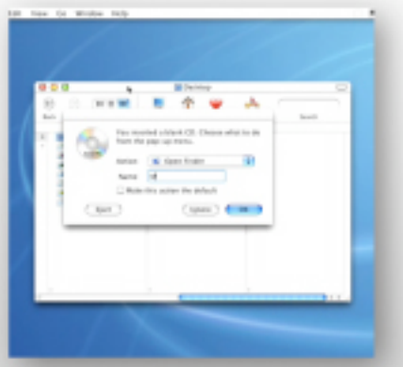
FireWire Reference Platform

Session 508





FireWire Reference Platform

Colin Whitby-Strevens
FireWire Architiect
colinws@apple.com

Agenda

Overview

System components

- TNF Kernel
- TNF Sys
- TNF Link Drivers

Protocols (two examples)

- TNF AV/C
- TNF SBP-2 Target

“TNF” = Thirteen-Ninety-Four

Overview—Summary

Source code and documentation for development of “end of the wire” FireWire devices

- Needs only a simple embedded RTOS
- Supports a variety of 1394 link chips
  - Several drivers provided—models for more
- Supports multiple FireWire application protocols
  - Simultaneously!
  - APIs for common FireWire application protocols

Overview—Contents 1

"Class Services"

- TNF Kernel
- AVC General
  - Disc, DVD, Panel and Tape subunits
- Asynchronous connections
- EIA 775 (on-screen display for DTV)
- IP1394 (RFC 2734—IPv4)
- PPDT
- SBP2

Overview—Contents 2

RTOS interfaces

- TNF sys
  - VxWorks
  - µiTRON
  - Nucleus
  - µCOS (early version)
  - Generic template

Board Support package

- x86 for VxWorks

Overview—Contents 3

Link drivers

- OHCI
- ceLynx
- GPLynx
- MD8412
- Generic link driver toolkit
- Plug-ins for hardware dependent features
  - GUID
  - DMA

Overview—Contents 4

Command line applications

- Zshell (interfaces command line applications)
- TMA (1394 transactions, view devices, bus reset, etc.)
- AVC Tape controller and target
- Bridge manager
- AVC Disc controller and target
- DVD controller and target
- PPDT
- SBP-2 disk
- etc.

Overview—Contents 5

Demos

- EIA 775
- PVR (2)
- IP1394
- Panel
- PVR Disc
- SBP2 Disk
- SBP2 Initiator
- etc.

Overview—Contents 6

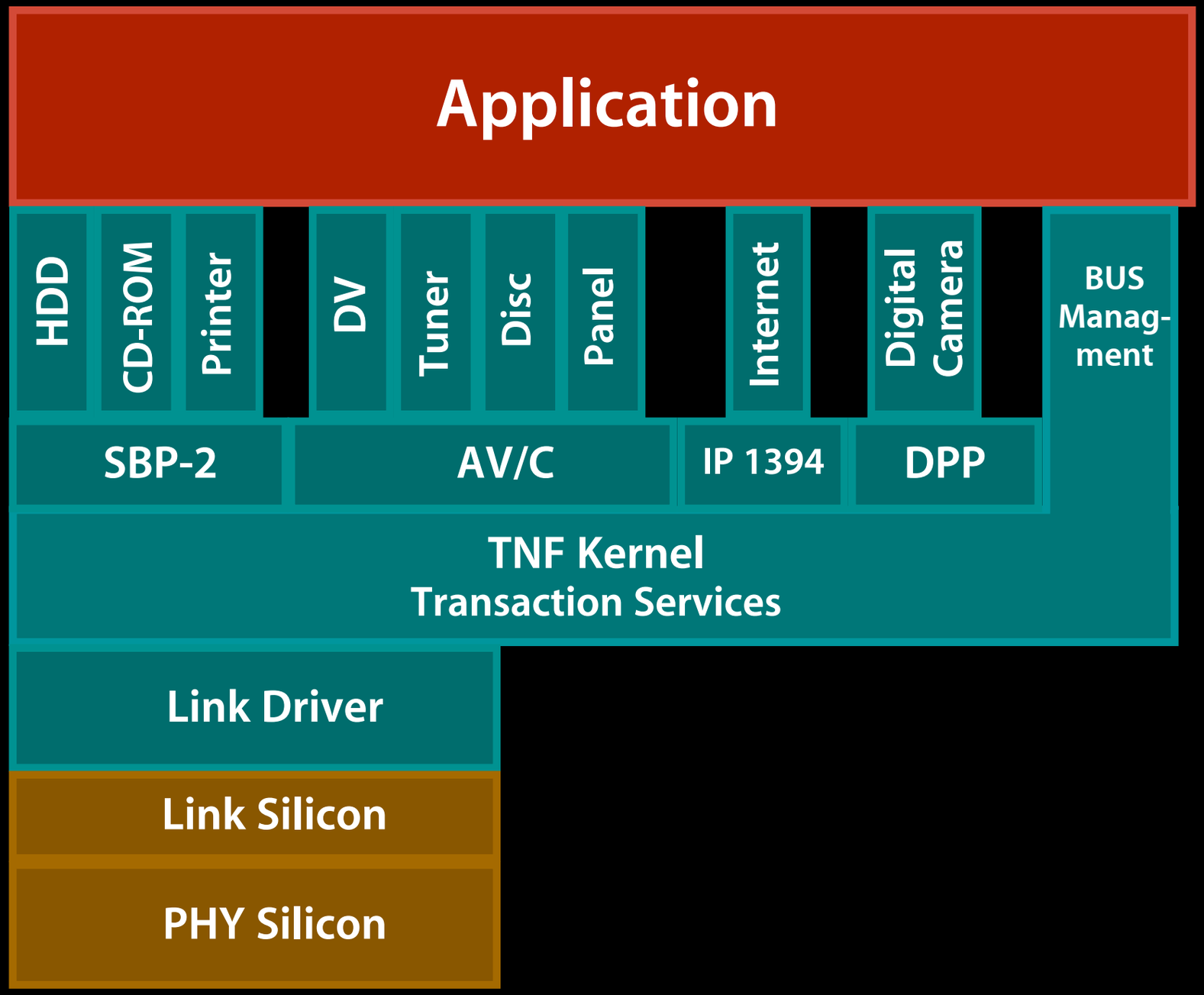
Documentation

- 20 pdf manuals
- release notes

Makefiles

Miscellaneous other tools

Overview—Architecture



TNF Kernel

TNF Kernel Functions 1

Incoming and outgoing transaction support

- Command and Indication queues
- Command and Response generation and handlers

Serial bus management:

- Bus Manager (BM) functions

TNF Kernel Functions 2

CSR Services

- Dynamic generation of configuration ROM

Reset services

- Handles topology changes
  - Automatically track other nodes by device reference

TNF Kernel Functions 3

Serial Bus Management

- Isochronous Resource Manager (IRM) function

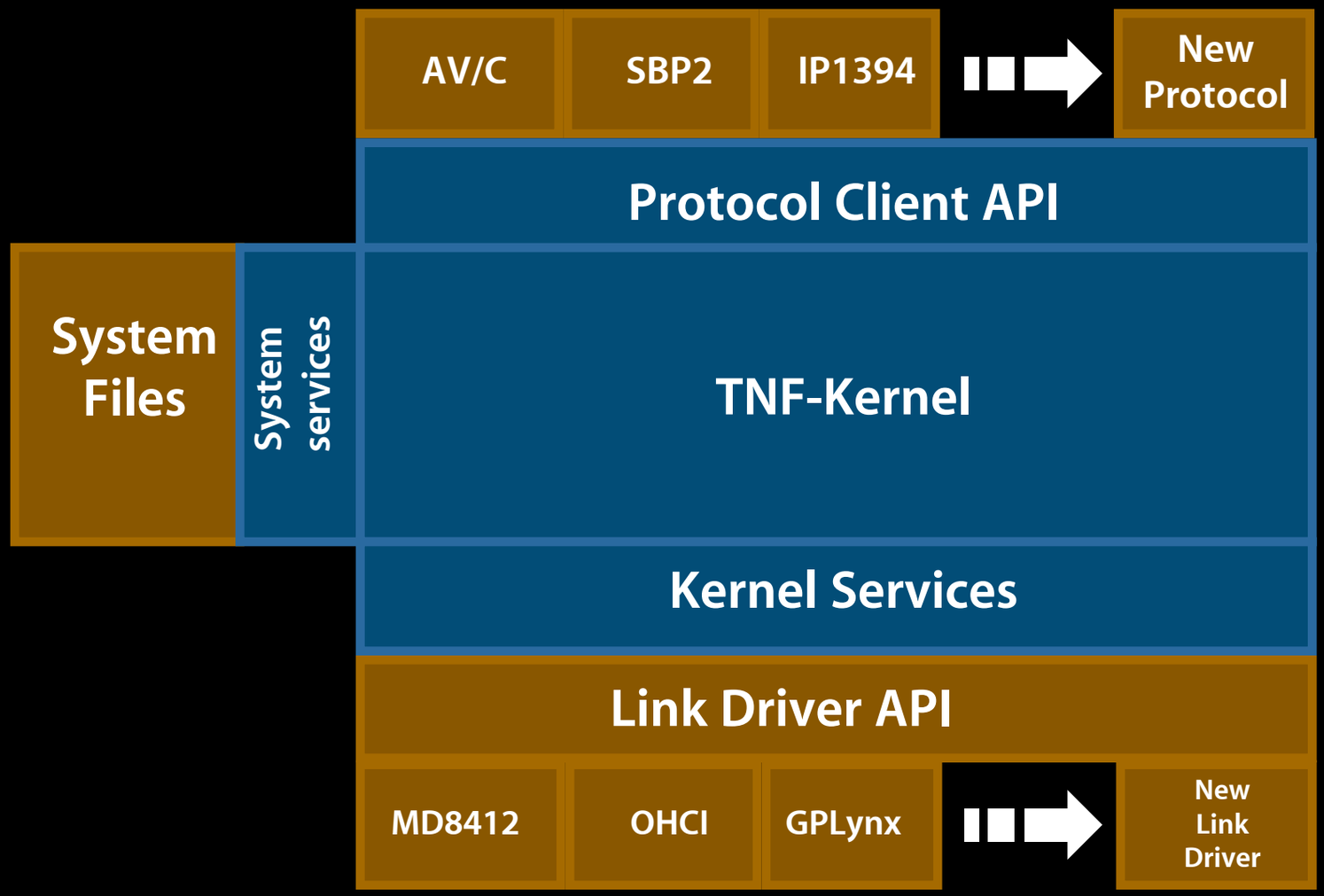
Function Control Protocol support

Isochronous Services

Output and Input Plug handlers

Where Is TNF Kernel Located?

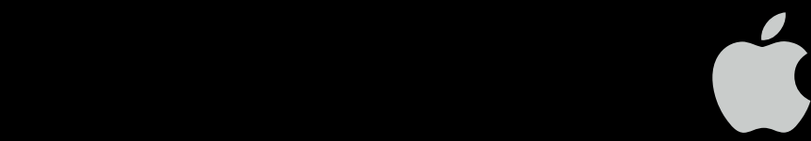
(Functional block diagram)



How Is TNF Kernel Accessed?

There are 3 primary Programming Interfaces (API)

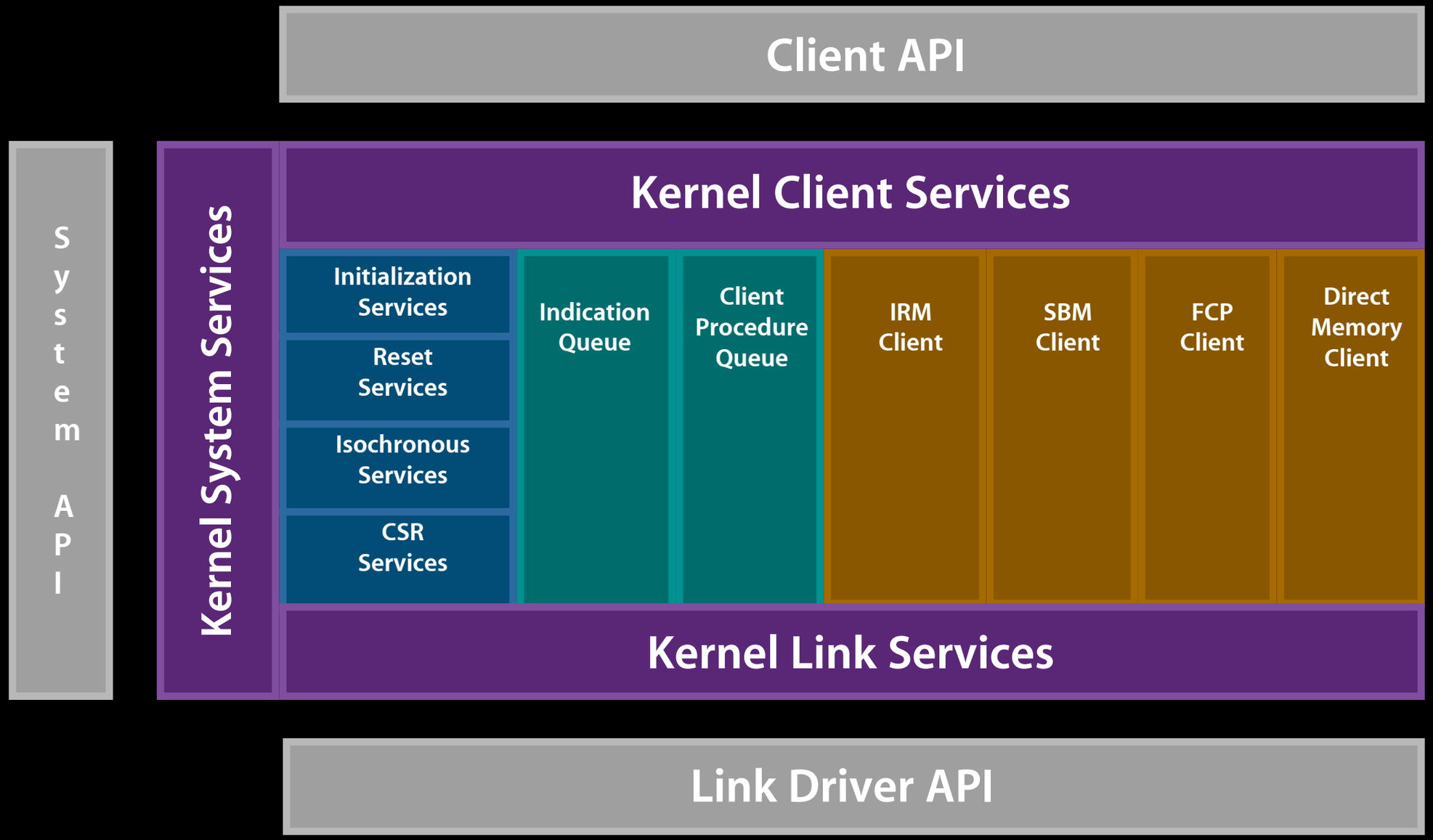
- Link Driver API (HAL layer interface)
  - Interface to hardware specific software driver
- System API
  - OS and system file interface
- Client API
  - Interface to all protocols and applications wishing to use TNF Kernel services to generate Serial Bus transactions



Why Use TNF Kernel?

- TNF Kernel provides stable programming interfaces that:
  - Abstract the 1394 serial bus from its clients (protocols and applications)
  - Isolate the hardware specific code to one concise driver
  - Allowing ease of portability to multiple CPUs and RTOSs

TNF Kernel Architectural Diagram



Link Driver API

Link driver API supports a wide variety of link devices

- Link capability function call is used to determine link capabilities
- Single command for bus subactions

Link driver plug-ins for specific hardware support

System Interface API

System calls isolated into one module (TNFsys.c) for easy of portability

Only 9 main system calls to support TNF Kernel tasking

- TNFSystemInitialize
- TNFCreateTask
- TNFCreateSemaphore
- TNFPostSemaphore
- TNFPendSemaphore
- TNFCreateQueue
- TNFDeleteQueue
- TNFSendQueueMessage
- TNFReceiveQueueMessage

Plus memory allocation, atomic operations etc

TNFsys.c is the common system API for all TNF software

Client Services

TNF Kernel supports an arbitrary number of clients

- Typical clients are transport protocols, applications, etc.
- Operate independently and in parallel
- Internal services (bus manager, IRM, FCP) implemented as clients

A new client acquires a Client ID from TNF Kernel

- Used in future calls

Client Services (Cont.)

A client can:

- Sign up for unit notification (for specific unit types that join the bus)
- Sign up for cable event notification (bus reset, etc.)
- Read CSR locations on 1394 nodes
- Search CSR ROMs on 1394 nodes for specific entries
- Use the kernel's other functions

Kernel Driver Services

A client needing to directly control bus devices gets a Device Reference ID

- Device identified initially by Global Unique ID

Unnecessary to remap 1394 node IDs after each 1394 bus reset

- Device reference remains the same

Driver services include

- Initiate all asynch request types
- Create isochronous stream descriptors
- Allocate isochronous channels and bandwidth
- Sign up to receive isochronous events
- Sign up for 1394 bus reset notifications

TNF Kernel Device Reference

Two ways of referencing a device on the bus

- Standard way
  - Uses GUID
  - TNF kernel automatically remaps after a bus reset
- Unspecified, for maximum control
  - Application requires to track the node ID and bus generation
  - Application will update the node ID of an unspecified device reference after a bus reset

Reset Services

Filters 1394 bus level reset from each client based on each clients need

- All bus resets
- Bus reset notification if a specific type device/unit joins the bus
- Bus reset notification if a specific type device/unit leaves the bus

Initialization Services 1

Kernel configuration

- Set the number of outgoing request queue commands
- Set the number of incoming requests that can be queued
- Number of client procedures (used with isochronous data streams) that can be queued

Understand the PHY and link's capabilities

Initialization Services 2

Node configuration

- Set OUI, busy retry code, ACK Pending control, etc.
- Optional features
  - Build bus registry (key CSR ROM entries from other nodes after bus reset)
  - Contend for bus manager
  - Optimize gap count
  - Isochronous capable
  - Power manager capable
  - Support dynamic CSR ROM
  - Externally provided CSR ROM
  - Cycle master function

PHY Services

Direct PHY Service client routines

- Helpful for bus managers or clients which need to have a low level knowledge of the bus
  - Force root
  - Send LinkOn
  - Set gap count
  - Ping node
  - Suspend/resume PHY port
  - Enable PHY port
  - Clear PHY port fault
  - Read/write remote PHY registers

CSR Services

Allows a client to prime the kernel to receive requests from remote nodes to read the local node's CSR ROM

- Functions to build the CSR ROM
- Modify or add units to the CSR ROM dynamically

Direct Memory Client

Direct memory client

- Allows access to the local node's physical memory

Notification control

- Notify never
- Notify every packet
- Notify n packets
- Notify n bytes

Optional support for posted writes

- Kernel sends write complete

Isochronous Services

Isochronous transaction routines

- Allocate/deallocate channels and bandwidth
- Initialize/start/stop streams

Isochronous event notifications, any combination of

- Start of isochronous data
- End of isochronous data
- Duplicate channel
- Unexpected channel
- Missing cycle start
- Late cycle start

FCP Services

FCP is essential for implementing an audio/visual control interface (specified in IEC 61883)

FCP functions:

- Send FCP command or response
- Add/remove FCP command handler (for incoming commands)
- Add/remove FCP response handler
- Add/remove FCP master output plug handler
- Add/remove FCP master input plug handler
- Add/remove FCP output plug handler
- Add/remove FCP input plug handler

Miscellaneous Features

Big/Little endian support

8/16/32 bit processor support

Compile time optimization for best possible footprint

Multi-threaded (multi-tasking) support

Using the TNF Kernel—Step By Step

Configure to determine queue sizes, etc.

Complete node configuration

- (whether bus manager capable, isoch resource manager capable, etc.)

Create CSR ROM

- TNF gets the capabilities of the Link driver and does a link driver configuration

Register for required services

Enjoy

TNF AV/C

TNF AV/C Summary

Provides an API for controllers and targets

- Support for multiple subunits
- Notification mechanism for bus and remote unit events
- Construction of conformant unit and subunit descriptors by targets
- Publication of AVC unit directory in ROM
- Parsing of unit and subunit descriptors by controllers
- The API provides references to units, subunits and plugs which remain unchanged thru bus-resets

TNF AV/C Features 1

Full support for v3.0 of the 1394 Trade Association AV/C General Specification

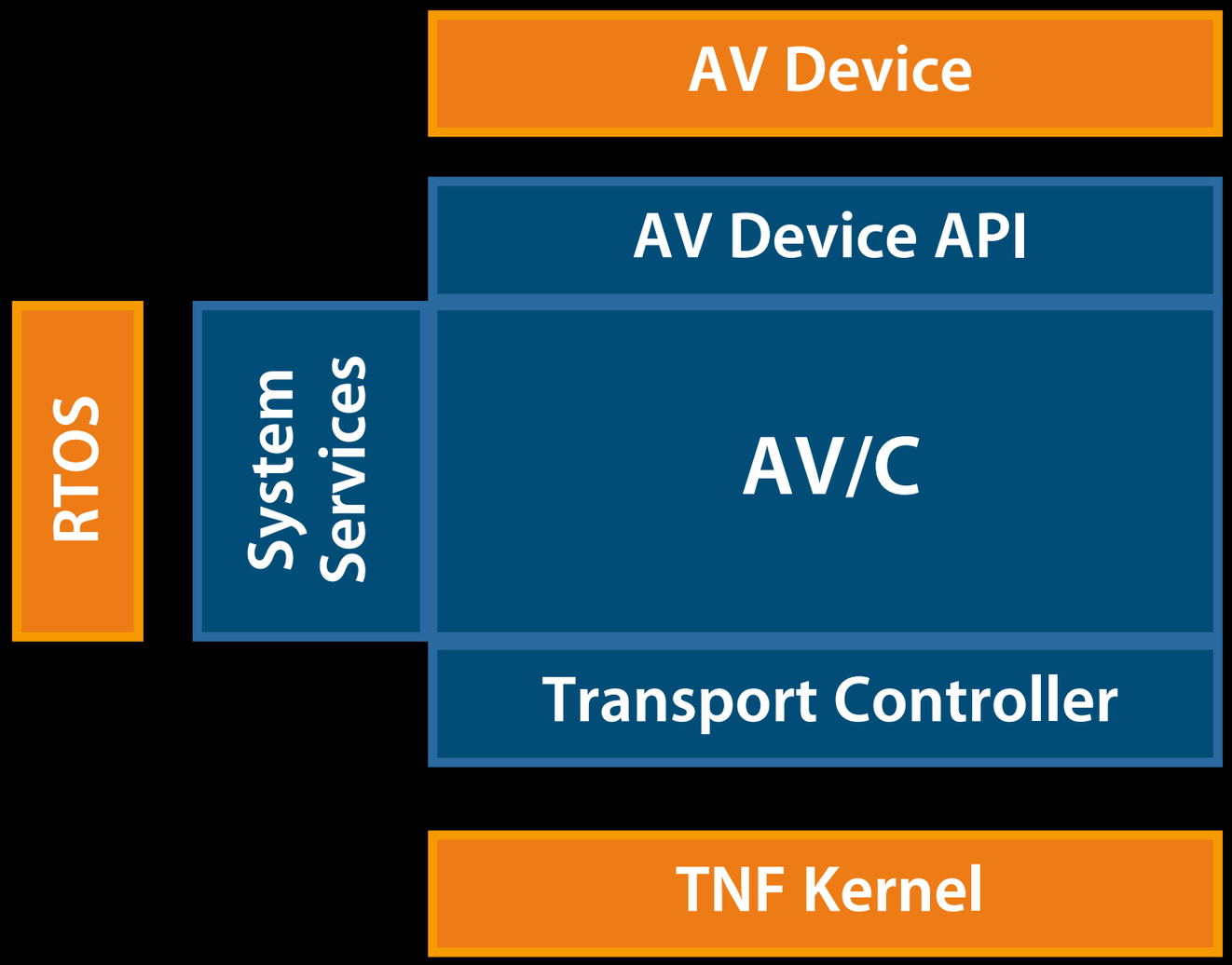
- Complete APIs for both controllers and targets
- Multiple sub-units
- Automatic mapping of addresses of units, subunits and plugs after bus reset
- Dynamic construction of descriptors (General Subunit Identifier, General Object List, General Object Entry)
- Ensures compliance with Open/Close rules for descriptors when changing state

TNF AV/C Features 2

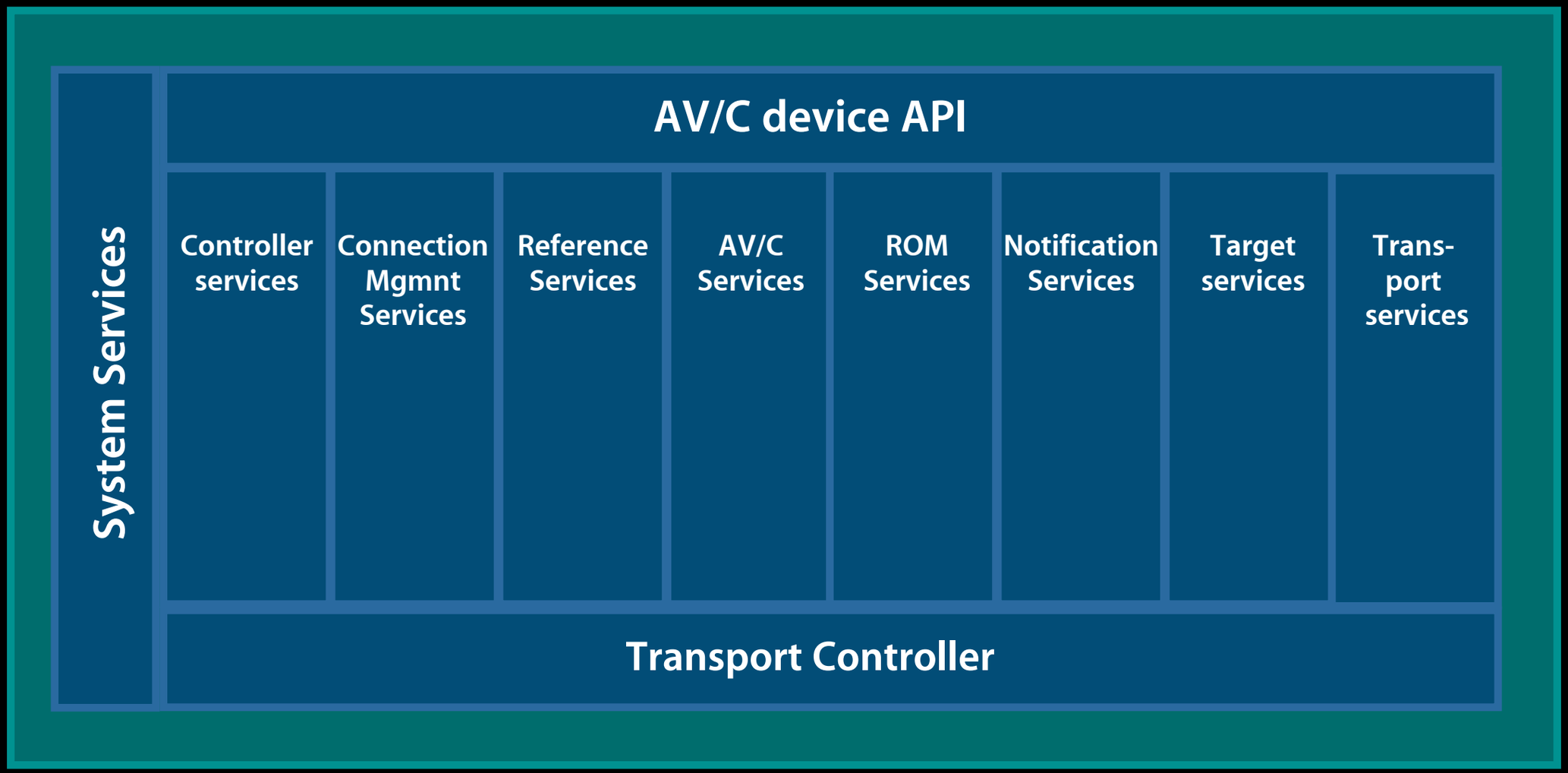
More features…

- IEC 61883 connection management
- FCP and/or Asynchronous connections for command transport
- Multiple simultaneous command transports of either type
- Multiple outstanding transactions
- Target can service multiple controllers

TNF AV/C Functional Block Diagram



TNF AV/C Architectural Diagram



System API

Main system requirement is to provide memory allocation

Error codes are defined in a header file

- Values may be operating system dependent

Transport Services

Supports multiple transports

Transports can be the same or different types

- Can support FCP and Asynchronous Connections simultaneously

Controller Services

Functions use AV Reference ID to identify the target device

- Send an AV/C command to a target
- Send a notify command
- Get unit info
- Get sub-unit info
- Get unit plug info

Connection Management Services

AV/C has its own connection management services providing logical plugs between subunits

- Very similar to IEC 61883 for 1394 devices

Connection Management Services (Cont.)

TNF AV/C connection management services:

- Configure output master plug
- Configure input master plug
- Connect/disconnect serial-bus plug
- Bring serial bus plug on-line/off line, transition on-line state
- Set/get serial bus plug configuration
- Register serial bus plug handler
  - Called on serial bus plug events
- Connect/disconnect between subunit plugs and other subunit plugs and/or unit plugs

Reference Services

All references to a device are made using a Device Ref ID

Device Ref ID provided as a result of exploration, or creation of local units/subunits

Services provide more information about a specific device

- Get AV/C ref type
- Get AV/C unit ref from AV/C ref
- Get AV/C subunit ref from AV/C ref
- Get AV/C plug ref from AV/C ref

AV/C Services

Initialize the AV/C device

- Controller, target or both

Register for unit event notification

Subunit address manipulation

Construct AV/C command header

Get plug ref

Target Services

Functions which are specific to AV/C targets

- Create a target unit
  - Returns an AV Unit Reference ID
- Create a target subunit
- Create an AV/C plug
- Register an AV/C command handler

TNF SBP-2 Target

ANSI NCITS 325

TNF SBP-2 Target

Provides full target support

- Generic APIs suitable for all types of SBP-2 targets

Fully compliant with ANSI NCITS 325

Operates as a client of TNF Kernel

- Uses only the TNF Kernel API, OS independent
- Will run in any environment in which TNF Kernel runs
- Call-back functions for CSR event handling

TNF SBP-2 Target (Cont.)

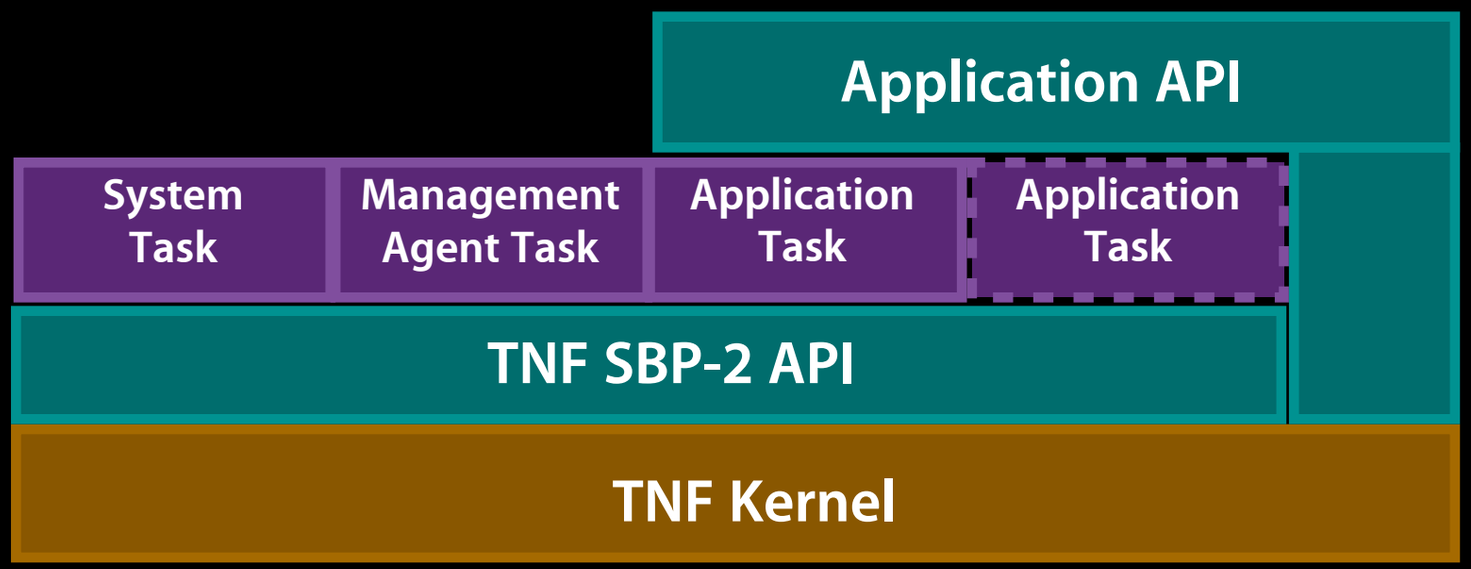
Application control of all SBP-2 parameters

- Full range of configuration functions
- Buffers, etc. allocated dynamically when initiator initiates each session

TNF SBP-2 Target System

TNF SBP-2 Target Components:

- System Task: Deals with CSR accesses
- Management Agent: Deals with initiator login etc.
- Application Task(s): Deals with ORB (Operation Request Block) processing



SBP-2 Compliance

Some aspects of SBP-2 compliance depend on appropriate application behavior

- Example: application is required to return status for each Operation Request Block (ORB) received
- This allows greater application flexibility
- Sample target application and documentation to assist compliance included in the TNF SBP-2 product

TNF SBP-2 APIs

API for initialization and termination

- Defaults for many configuration parameters
- Application normally provides configuration information as a data structure at initialization time

APIs for configuration and control functions

- allows dynamic management by the application of configuration parameters

Status block APIs to send status back to the initiator

Data transfer APIs

Access to TNF Kernel

Application has full access to the TNF Kernel API

May be useful for functions such as

- Get Target Node ID
- Get Target EUI64

TNF Kernel API provides application isolation from the RTOS

- TNFmalloc, TNFstrcpy, TNFmemcpy, etc.

Data Transfer APIs

Automatic support for initiator page tables

- API deals internally with normalized pages tables and unrestricted page tables

Application calls StartDataTransfer

- Determines whether page table or direct memory is being used
- Reads page table if necessary, initializes a segment array for the application to use in accessing data
- Optional verification of initiator page table

Data Transfer APIs (Cont.)

Application makes repeated calls to ReadBuffer or WriteBuffer

- Identifies buffer, identifies segment index if page table
- Support for error type reporting

Application calls StopDataTransfer

Status APIs

Send Unsolicited Status

Send Command Request Complete Status

- Maybe with an error indication

Send Command Error Status

- Also causes Command Agent to transition to dead state

Notify Orb Error Status

- Causes Command Agent to transition to dead state
- Must be followed by either Request Complete or Command Error Status, as the application must ensure that a Status block is returned for each ORB

Event Handlers Provided By the Application

Event handlers may be called when the initiator reads/writes to SBP-2 CSRs in the target

- Management Agent CSR
- Command Agent CSRs
- Reset Start CSR

Some initiator reads/writes are handled entirely within TNF SBP-2, and no application event handler is needed

- Example: query login

Event Handlers Provided By the Application (Cont.)

Event handler for work to do (ORBs to process)

Event handler for error conditions and Bus Reset

Some event handlers are optional

- Provided only if the application cares

Configuration of Callbacks

Application calls SBP2TgtInit with one callback parameter: OnOpenSession

TNF SBP-2 Management Agent calls OnOpenSession for each login request

Management Agent provides the login descriptor as a parameter to OnOpenSession

- Used to maintain all the information related to the login

Entries for all the other callbacks are contained within this data structure

- Allows callbacks to be specified on a per-login basis

OnOpenSession initializes these entries

Application Callback Functions

Called as a result of reads/writes to Management Agent

- OnOpenSession
  - Called on login (i.e., when the initiator initiated a new session)
  - Allocates the CSR space for command registers
  - Sets up the other call-backs
  - Allocates the ORB buffer(s)
- OnEndSession
  - Abort all tasks, free memory
- OnSetPassword
  - Store new password in non-volatile RAM
- OnAbortTaskSet
- OnResetTarget (optional)

Application callback Functions

Called as a result of reads/writes to Command Agent

- OnORBBufferFull
  - ORBs are ready for processing
  - Also called when the end of the ORB linked list is reached
  - ORB processing should be performed by a separate task to free up this task to receive more messages
  - Send message to ORB processing task
- OnDoorbell (optional)
  - Doorbell processing is self contained within the Command Agent
  - But the application may wish to take some action too, such as allocating more buffers

Application callback Functions

Called as a result of internal conditions

- OnAgentDead
  - Command agent has died (probably indicates a more serious problem with the bus or the initiator)
  - Application has to abort all task sets, may also try to send unsolicited status to the initiator
- OnBusReset
  - Application has to abort all task sets
- OnUnitAttention
  - Send unsolicited status to all logged in initiators
  - May add command-set specific information to the status message

More Information

FireWire Reference Platform

http://developer.apple.com/firewire/platform.html

Mailing Lists

http://www.lists.apple.com/mailman/listinfo/firewire-reference-platform/

firewire-reference-platform@lists.apple.com

Q&A

Prashant Kanhere
Program Manager—FireWire Technologies
Prashant@apple.com

http://developer.apple.com/wwdc2003/urls.html

WWDC 2003

Worldwide Developers Conference